Insight:

Scopes

There are 3 keywords that change the current scope in ruby code.
They are: class, module and def.

Our progress so far:

- ~~Open classes~~
- ~~Duck Typed~~
- ~~Just a runnable code~~
- ~~Object themselves~~*
- Scopes
- Modules with hierarchies
- Inheritance hierarchy
- Ruby object model
- Method lookup
- Etc.

What is Scope?

A virtual environment where variables and methods live.

If you are in the scope you can access those variables and methods. If you are not in the scope, or in other words, out of scope, you can’t see or call those methods and variables.

Accessing scope

- Accessible by “self” pseudo variable
- “self” is always available and points to the current scope
- Instance variables and methods without explicit receiver are looked up in the current scope.

Calling method on explicit receiver

```ruby
class MyClass
    def foo
        bar                                     # This is read as `self.bar()`
        String.try_convert("str")               # Calling "try_convert" on explicit receiver "String"
    end

    def bar
        @instance_var                           # @instance_var is searched in `self`
    end
end

obj = MyClass.new
obj.foo                                         # Method `foo` is called with explicit receiver `obj`
```

Quick Recap

- Instance variables are searched in current scope
- Methdos without explicit receiver object are called on current "self"

There are 3 keywords that change scope

1. class
2. module
3. def

Values of “self”

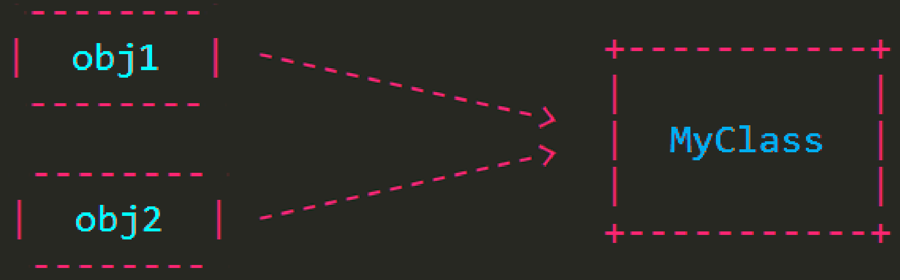
```
puts self                                                # main

class MyClass
    puts self                                            # MyClass

    def instance_method
        puts self
    end

    puts self                                            # MyClass
end

puts self                                                # main

obj1 = MyClass.new                                       # #<MyClass:0x007fde2426caa0>
obj2 = MyClass.new                                       # #<MyClass:0x007fde24294370>

obj1.instance_method                                     # #<MyClass:0x007fde2426caa0>
obj2.instance_method                                     # #<MyClass:0x007fde24294370>
```

Scope changes

- Top-level scope:
    - Instance of Object class
    - Always named "main"
- Within class definition, changes to the class itself
- Within "def" block, changes to an instance of the class

Visualizing it

```
puts self                                       # main

class MyClass
    puts self                                   # MyClass

    def instance_method
        puts self
    end

    puts self                                   # MyClass
end

puts self                                       # main

obj1 = MyClass.new                              # #<MyClass:0x007fde2426caa0>
obj2 = MyClass.new                              # #<MyClass:0x007fde24294370>

obj1.instance_method                            # #<MyClass:0x007fde2426caa0>
obj2.instance_method                            # #<MyClass:0x007fde24294370>
```



Values of “self” and eigenclasses

```
class MyClass
    class << self
        puts self                                       # #<Class:MyClass>

        def class_method
            puts self
        end
    end

    def self.class_method_2
        puts self
    end
end

class << MyClass
    puts self                                           # #<Class:MyClass>
end

MyClass.class_method                                    # MyClass
MyClass.class_method_2                                  # MyClass
```

Scope changes

- Top level – instance of Object class
- Within class definition – class itself
- Within “def” block – instance of the class
- “class << ClassName” – eigenclass scope
- Class methods – class itself

Our progress so far:

- ~~Open classes~~
- ~~Duck Typed~~
- ~~Just a runnable code~~
- ~~Object themselves~~*
- ~~Scopes~~
- Modules with hierarchies
- Inheritance hierarchy
- Ruby object model
- Method lookup
- Etc.